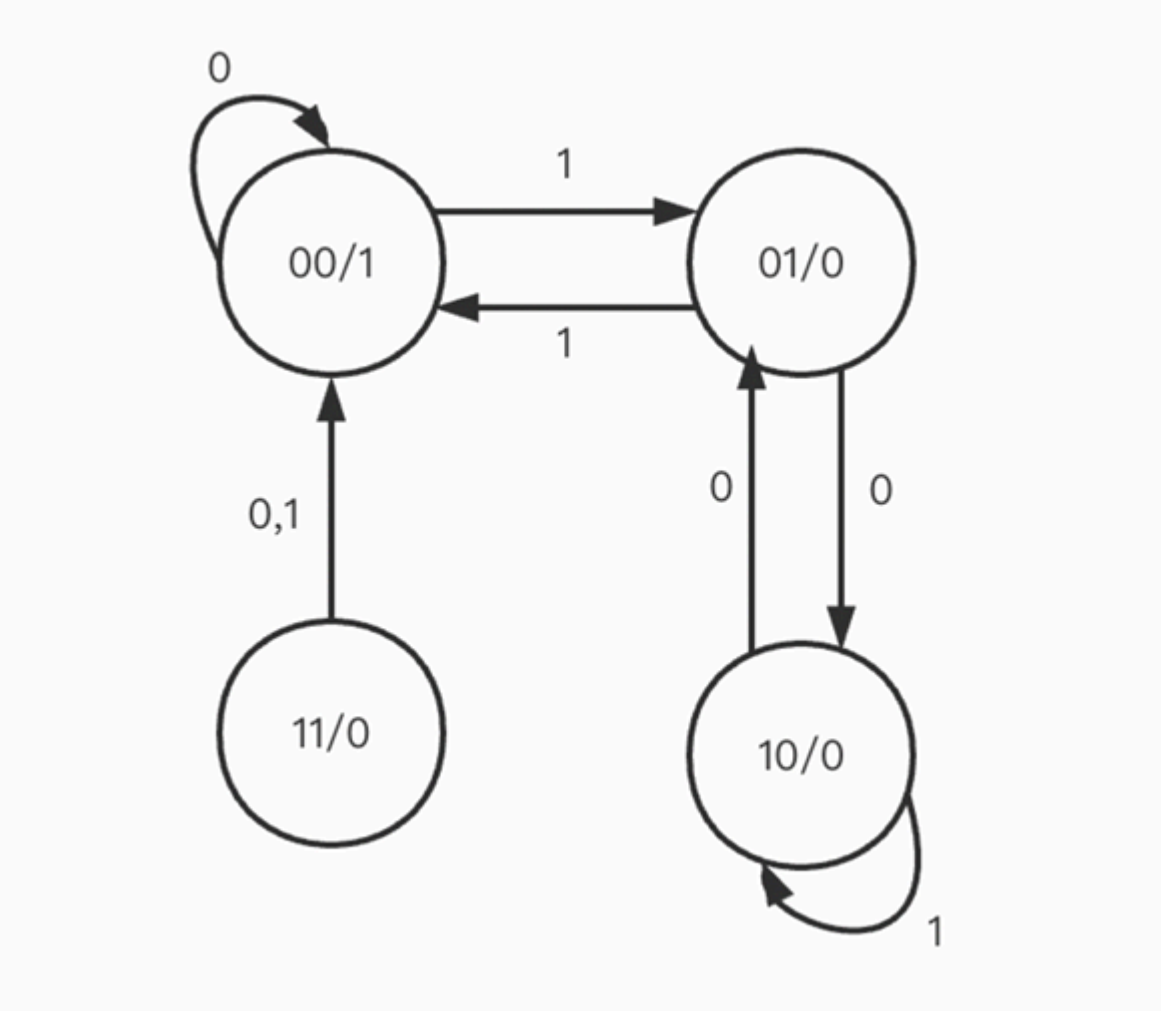
T1

(1).

S_1	S_0	X	Z	S_1'	S_0'
0	0	0	1	0	0
0	0	1	1	0	1
0	1	0	0	1	0
0	1	1	0	0	0
1	0	0	0	0	1
1	0	1	0	1	0
1	1	0	0	0	0
1	1	1	0	0	0

(2).



T2

前 4 个都比较显然,查书即可，第 5 个注意循环 R0 循环加 2 直到加到 x8000后循环中止，再加上后面的 3 和 1，得到 x8004.

X	Does the program halt?	Value stored in `R0`
000000010	Yes	2
000000001	Yes	3
000000000	Yes	6
111111111	No	--
111111110	Yes	x8004 ($-2^{15}+4$)

T3

FETCH: 1 + 100 + 1 = 102
DECODE: 1 (memory access is not required)
EVALUATE ADDRESS: 0 (this phase is not required)
FETCH OPERANDS: 1 (memory access is not required)
EXECUTE: 1 (memory access is not required)
STORE RESULT: 0 (this phase is not required)

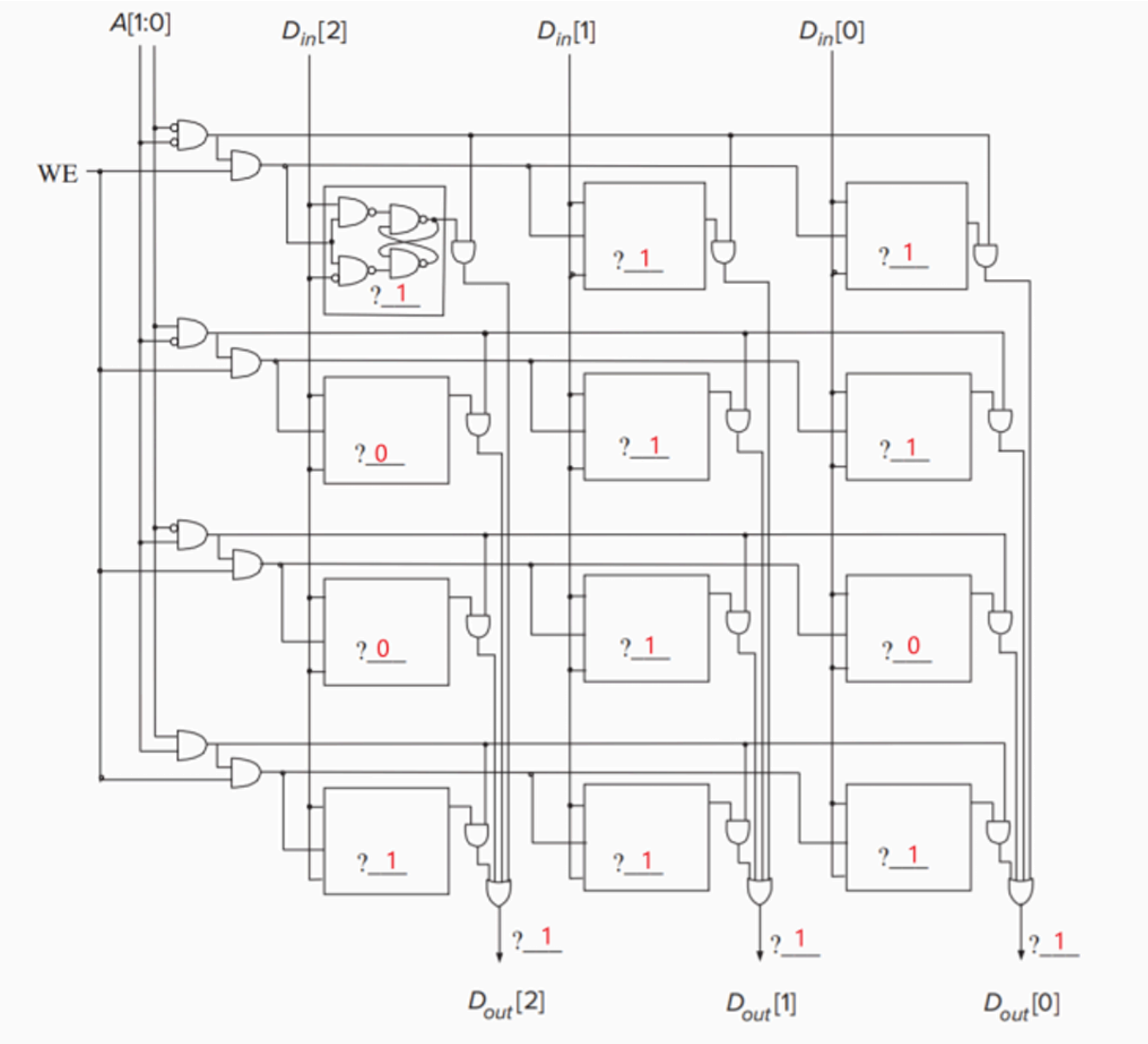
T4

1. ADD(0001) 和 AND(0101) 有了更大的立即数范围。但 NOT(1001) 没有获益。
2. LD(0010) 和 ST(0011) 能有更多的一位去寻址。
3. BR(0000) 中没有寄存器，因此没有获益。

T5

a. OPCODE 有 225 条，需要 8 位才能表示完整
b. 寄存器有 120 个，需要 7 位才能表示完整
c. 32 - 7 * 3 - 8 = 3，最多有 3 位 UNUSED

T6



只要看每个地址最后一次被写时Din的值即可,
则地址 00 应该是 cycle 6 写的 111, 01 是 cycle 4 写的 011, 10 是 cycle 3 写的 010, 11 是 cycle 7 写的 111, 看最后一个 cycle, 在读 11 处的值, 所以读出 111.

T7

此题需要严密的推理。

Operation No.	R/W	MAR	MDR
1	W	x4000	11110
2	R	x4003	10110
3	W	x4001	10110
4	R	x4002	01101
5	W	x4003	01101

Operations on Memory

Address	Before Access 1	After Access 3	After Access 5
x4000	01101	11110	11110
x4001	11010	10110	10110
x4002	01101	01101	01101
x4003	10110	10110	01101
x4004	11110	11110	11110

Contents of Memory locations

为了叙述方便, 下面会简称 Operation i 为 i, 表 1 指代 Operations on Memory, 表 2, 3, 4 指代 Memory before Access 1, Memory after Access 3, Memory after Access 5. 同时建议大家在看的同时按照我的推导填表. 因为The data in the MDR must come from a previous read (load). 而 Operation 3 的 MDR 与 1 有不同, 说明 Operation 2 是 Read. 而 Memory 在 Access 3 后与 Access 1 前 x4000 和 x4001 都发生了变化, 说明 Operation 1 和 3 写了这两处地址, 假设 1 写了 x4001, 与第二位是 0 矛盾, 因此 1 写了 x4000, 3 写了 x4001, 同时该步的 MDR 是 10xx0, 比对表 2, 知道应该来自 x4003. 到这里填出了表 1 前 3 行和表2, 3 除了 x4002 外的行.之后发现 x4003 在 Access 5 由 10110 变成了 01101, 发生了 Write, 说明 4 是 R, 5 是 W, 5 的 MDR 是 01101, 考虑到表 3 在上一步填完后没有 01101, 说明 x4002 处值为 01101, 得解, 最终答案如上。

T8

以下用 $a \oplus b$ 表示 a XOR b，$a + b$ 表示 a OR b，$a \cdot b$ 表示 a AND b，$\bar{a}$ 表示 NOT a，则有 $a \oplus b = a \cdot \bar{b} + \bar{a} \cdot b = \overline{\overline{a \cdot \bar{b} + \bar{a} \cdot b}} = \overline{\overline{a \cdot \bar{b}} \cdot \overline{\bar{a} \cdot b}}$.

Address	Instruction	Comments
x3000	**1001 111 001 111111**	NOT R7, R1
x3001	1001 110 010 111111	NOT R6, R2
x3002	0101 101 111 000 010	AND R5, R7, R2
x3003	**0101 100 110 000 001**	AND R4, R6, R1
x3004	1001 001 101 111111	NOT R1, R5
x3005	**1001 010 100 111111**	NOT R2, R4
x3006	**0101 000 001 000 010**	AND R0, R1, R2
x3007	1001 011 000 111111	NOT R3, R0

T9

其中第2条可以作为NOP，因为其偏移设置为0

1. 将R1中的值与立即数2相加并存放到R2中
2. NOP
3. 检查标志位N和P，若其中一个被置位则跳转到对应的位置
4. 对R7求反存放到R2中
5. TRAP指令，读取内存单元x0023中的内容作为服务程序入口

T10

BR指令无法跳转到PC寄存器和偏移量之和范围以外的指令，即其最大范围为距离当前指令+256到-255的地址空间
对于JMP指令，其能够将对应寄存器的内容装入PC寄存器，使得程序执行流可以跳转至内存空间的任意位置